Rust for Rubysts

Filipe Costa

@filipebarcos


HOTEL
BEIRAMAR

ride

#ONELESSCAR

What is Rust?

Systems Programming Language

“language used for system programming; such languages are designed for writing system software”

https://en.wikipedia.org/wiki/System_programming_language

“System software is computer software designed to operate and control the computer hardware, and to provide a platform for running application software”

https://en.wikipedia.org/wiki/System_programming_language


mozilla

Servo

github.com/rust-lang/rust

github.com/servo/servo

Characteristics

Compiled

Strong Statically Typed

Really Good Concurrency Support

I cannot write *wrong* concurrent code

Very Very Fast

No GC

Memory Safe

Type Inference

Rust

Stack x Heap Allocation

Stack

Address	Name	Value

Heap

Address	Name	Value

Stack

Address	Name	Value

```
let foo = 1;
```

Heap

Address	Name	Value

Stack

Address	Name	Value
0x00	foo	1

Rust

```
let foo = 1;
```

Heap

Address	Name	Value

Stack

Address	Name	Value
0x00	foo	1

Rust

```
let foo = 1;
```

```
let bar = 9;
```

Heap

Address	Name	Value

Stack

Address	Name	Value
0x01	bar	9
0x00	foo	1

Rust

```rust
let foo = 1;
let bar = 9;
```

Heap

Address	Name	Value

Stack

Address	Name	Value
0x01	bar	9
0x00	foo	1

Rust

```rust
let foo = 1;
let bar = 9;
```

```
foo = 1
```

Heap

Address	Name	Value

Stack

Address	Name	Value
0x01	bar	9
0x00	foo	1

Rust

```
let foo = 1;
let bar = 9;
```

Ruby

```
foo = 1
```

Heap

Address	Name	Value
		1
0x00	foo	

Stack

Address	Name	Value
0x01	bar	9
0x00	foo	1

Rust

```
let foo = 1;
let bar = 9;
```

Ruby

```
foo = 1
```

Heap

Address	Name	Value
0x08		1
0x00	foo	0x08

Stack

Address	Name	Value
0x01	bar	9
0x00	foo	1

Rust

```
let foo = 1;
let bar = 9;
```

```
bar = 9
```

Ruby

```
foo = 1
```

Heap

Address	Name	Value
0x08		1
0x00	foo	0x08

Stack

Address	Name	Value
0x01	bar	9
0x00	foo	1

Rust

```rust
let foo = 1;
let bar = 9;
```

Ruby

```ruby
foo = 1
bar = 9
```

Heap

Address	Name	Value
0x08		1
		9
0x01	bar	
0x00	foo	0x08

Stack

Address	Name	Value
0x01	bar	9
0x00	foo	1

Rust

```rust
let foo = 1;
let bar = 9;
```

Ruby

```ruby
foo = 1
bar = 9
```

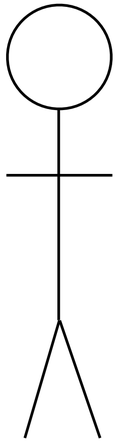
Heap

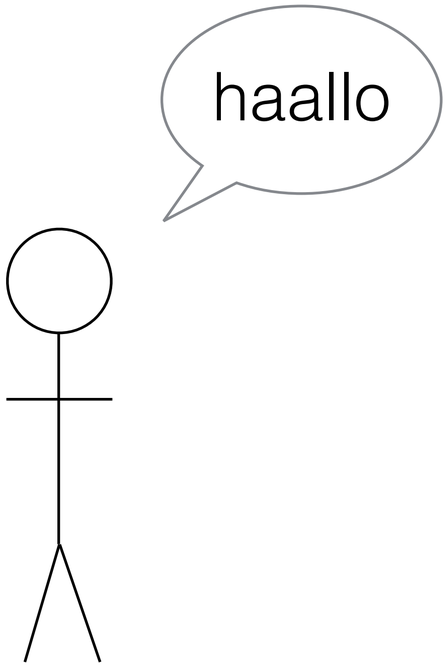
Address	Name	Value
0x08		1
0x07		9
0x01	bar	0x07
0x00	foo	0x08

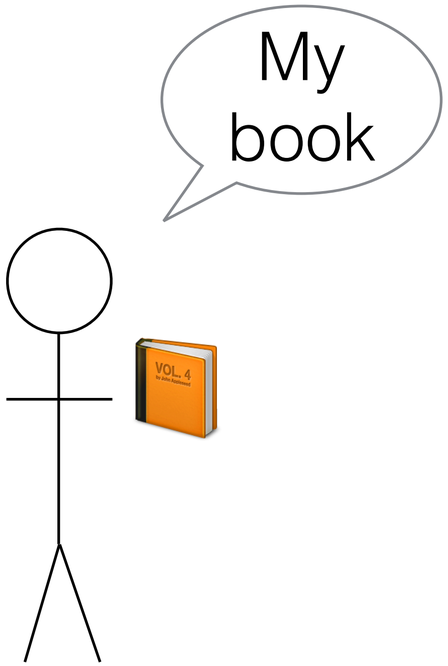
Rust has Heap Allocation

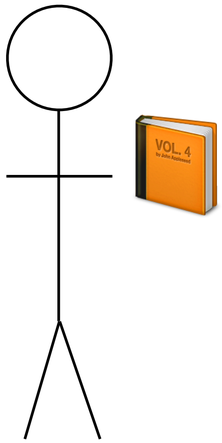
```
let foo = Box::new(5);
```

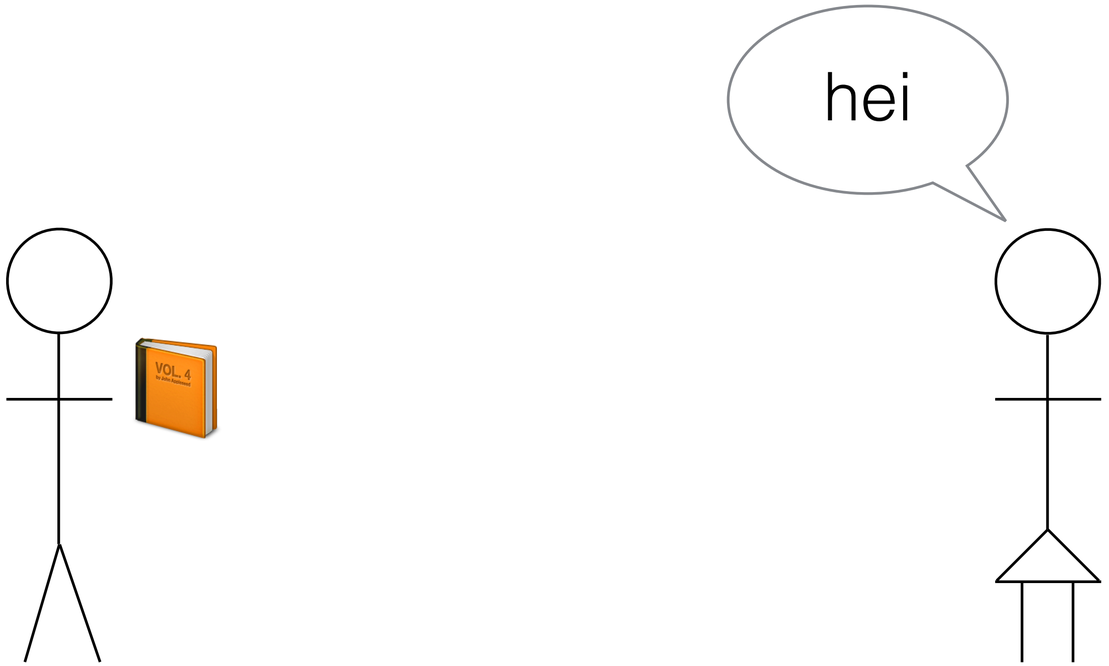
Ownership

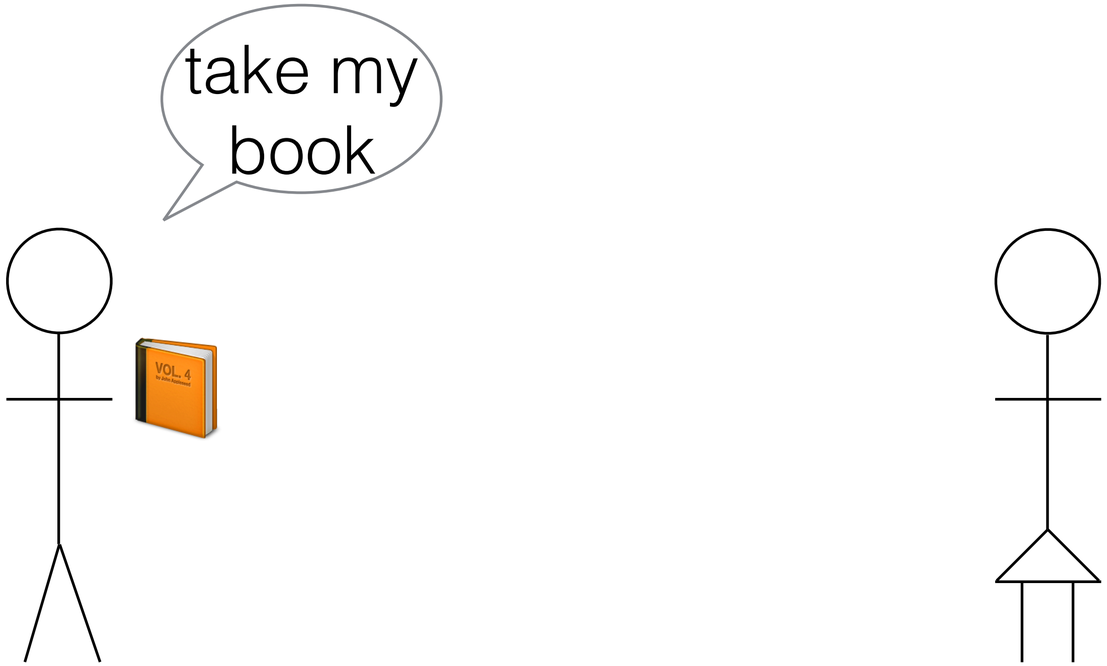


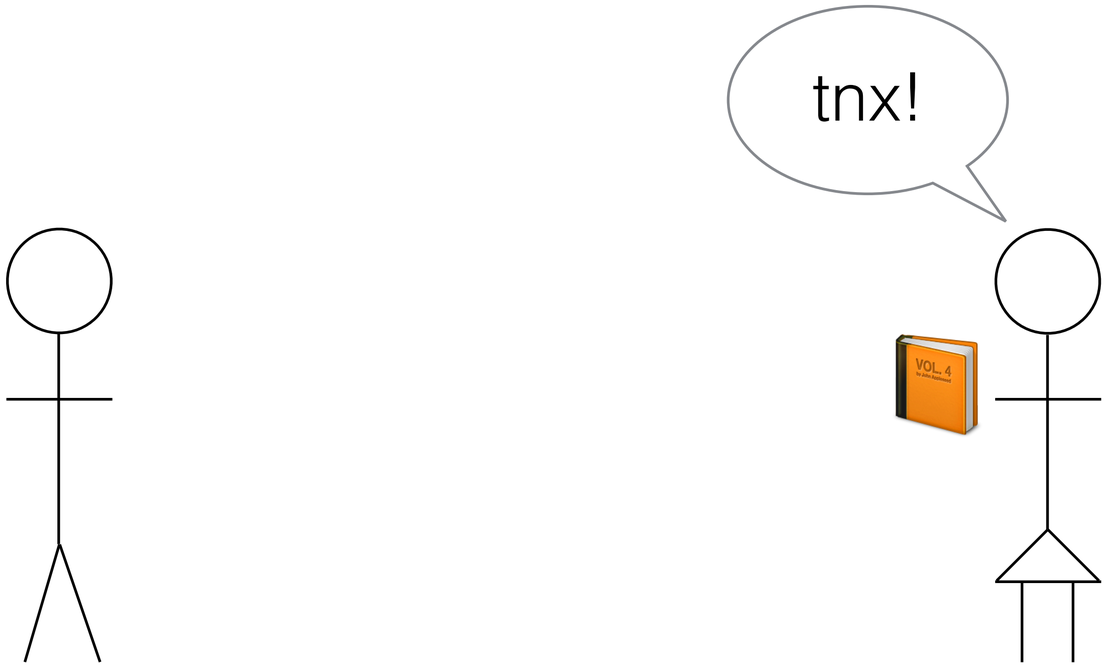

haallo


My
book
VOL. 4


VOL. 4


hei
VOL. 4


take my book
VOL. 4


tnx!
VOL 4

Ownership

```rust
fn main() {
    let number = Box::new(3);
    helper(number); //moves the value!
    helper(number); // error!
}

fn helper(number: Box<i32>) {
    println!("Number was: {}", number);
}
```

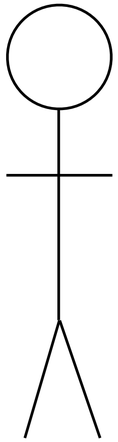
Ownership

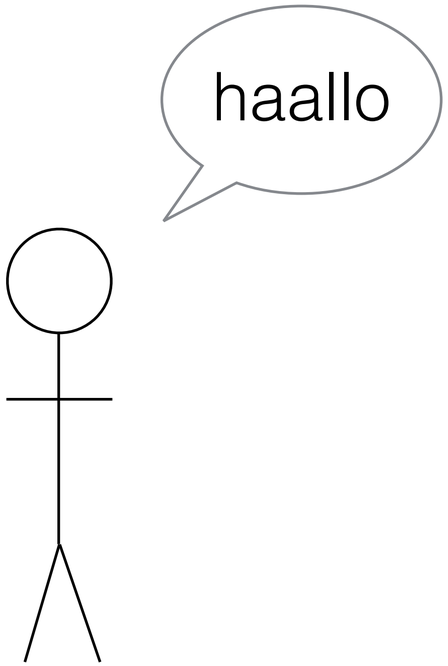
```
ownership.rs:4:12: 4:18 error: use of moved value: `number` [E0382]
ownership.rs:4     helper(number); // error!
                          ^~~~~~
ownership.rs:3:12: 3:18 note: `number` moved here because it has type `Box<i32>`,
which is non-copyable
ownership.rs:3     helper(number); //moves the value!
                          ^~~~~~
error: aborting due to previous error
```

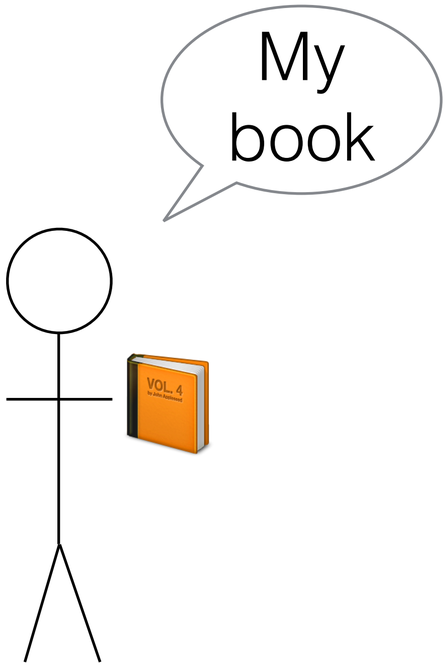
Ownership

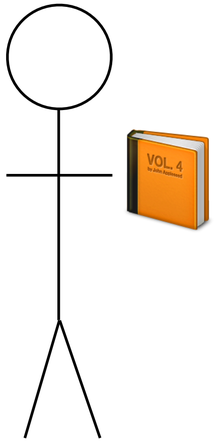
```
ownership.rs:4:12: 4:18 error: use of moved value: `number` [E0382]
ownership.rs:4     helper(number); // error!
                         ^~~~~~
ownership.rs:3:12: 3:18 note: `number` moved here because it has type `Box<i32>`,
which is non-copyable
ownership.rs:3     helper(number); //moves the value!
                         ^~~~~~
error: aborting due to previous error
```

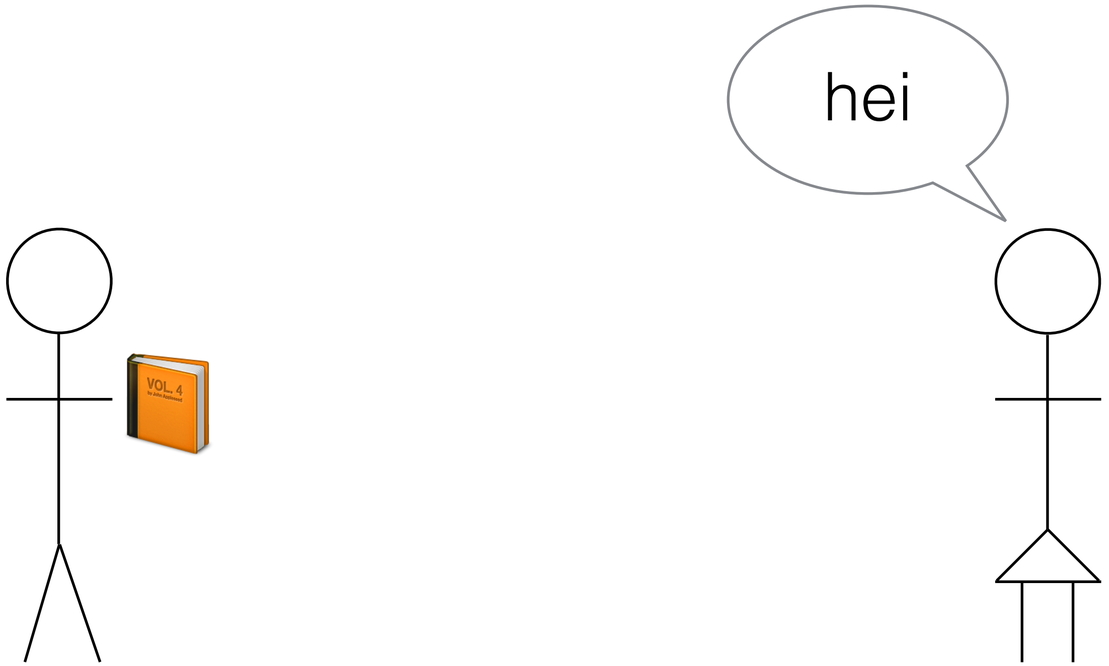
Borrowing

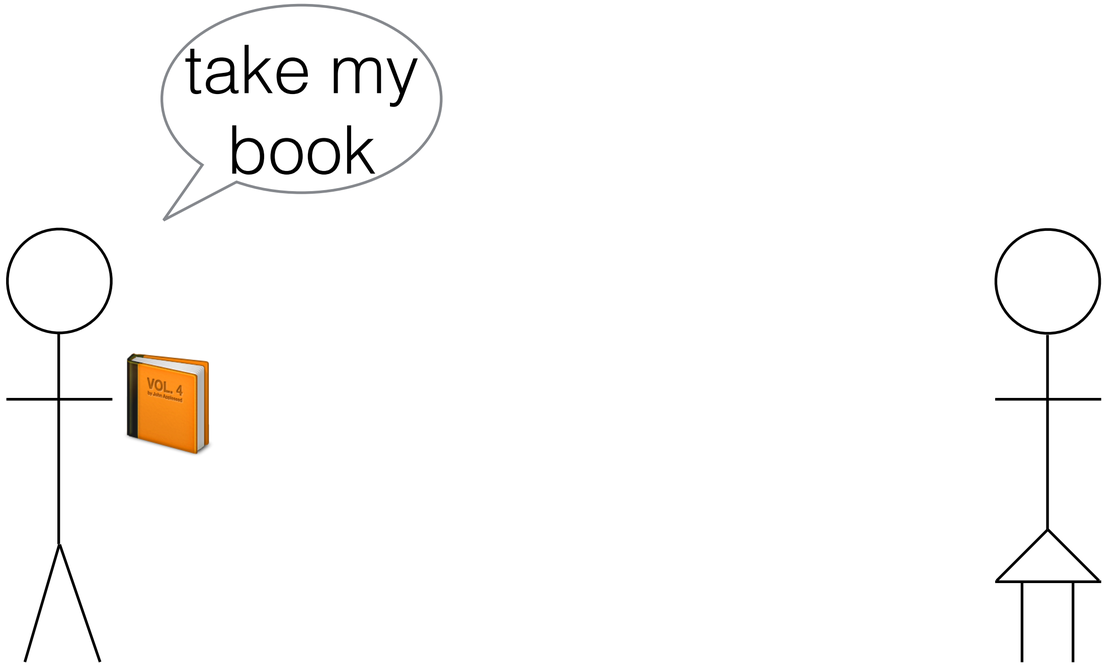


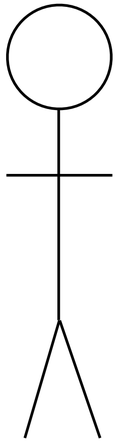

haallo

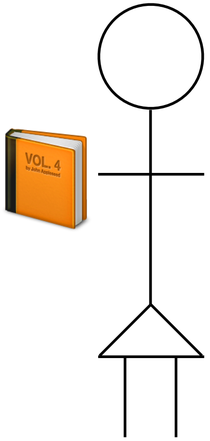

My
book
VOL. 4

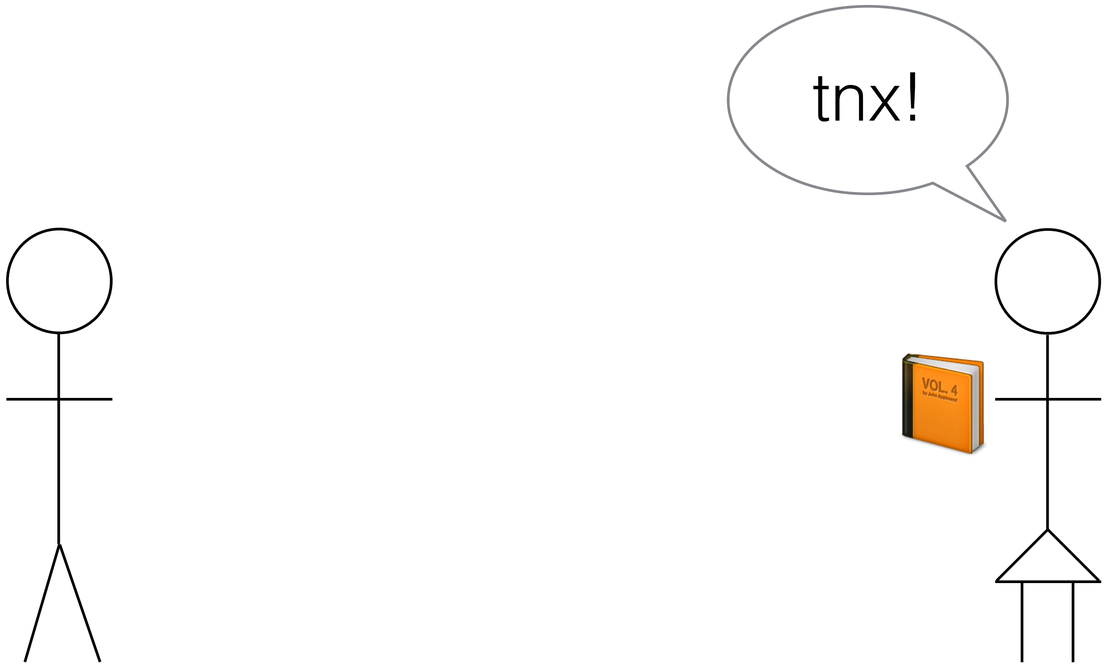

VOL. 4

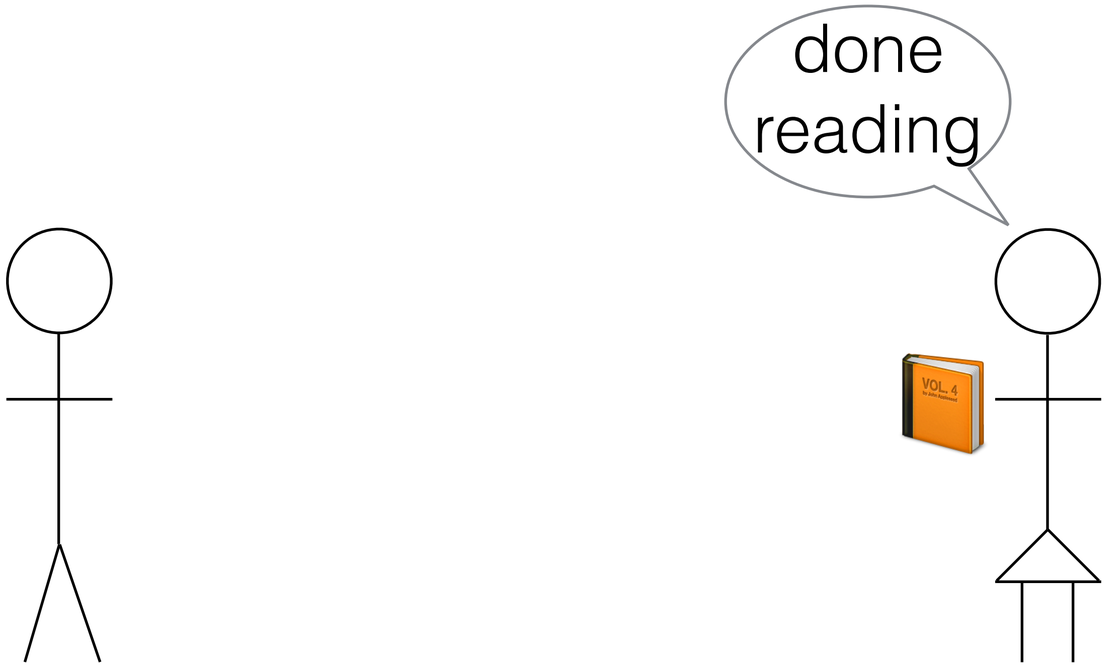

hei
VOL. 4

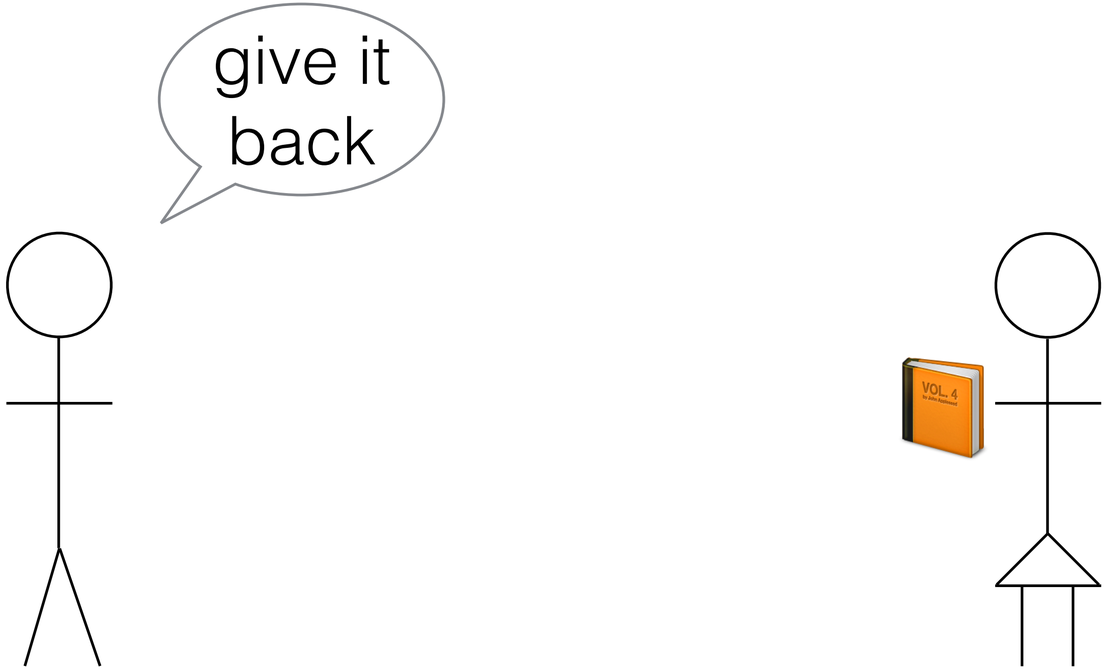

take my book
VOL. 4

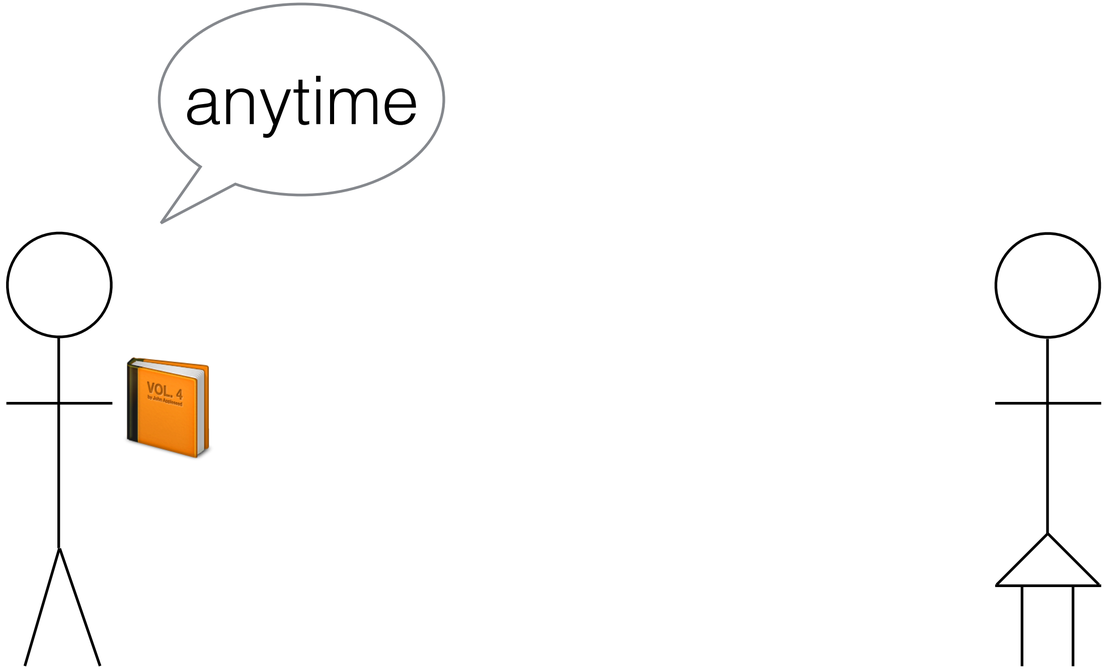



VOL. 4


tnx!
VOL. 4


done
reading
VOL. 4


give it back
VOL. 4


anytime
VOL. 4

Borrowing

```rust
fn main() {
    let number = Box::new(3);
    helper(number); //moves the value!
    helper(number); // error!
}

fn helper(number: Box<i32>) {
    println!("Number was: {}", number);
}
```

Borrowing

```rust
fn main() {
    let number = Box::new(3);
    helper(&number);
    helper(&number);
}

fn helper(number: &Box<i32>) {
    println!("Number was: {}", number);
}
```

Borrowing

```
fn main() {
    let a: &i32;
    {
        let b = 3;
        a = &b; // error!
    }
}
```

Borrowing

```rust
fn main() {
    let a = Box::new(1);
    let b = &a;
    helper(a);
}

fn helper(i: Box<i32>) {
    // something
}
```

Mutability

Mutability

```
fn main() {
    let mut a = 1;
    a = 4;
}
```

Mutability + Borrow

```rust
fn main() {
    let mut a = 1;
    double(&mut a);
    println!("{}", a);
}

fn double(i: &mut i32) {
    *i *= 2;
}
```

Rust x Ruby

Classes Vs. Structs

```ruby
# ruby
class Person
    attr_reader :name
end
person = Person.new(name: ‘Filipe Costa’)
```

```rust
// Rust
struct Person {
    name: String
}
let person = Person { name: "Filipe Costa" };
```

Functions

```
fn add(x, y) {
    x + y;
}
```

Functions

```
fn add(x, y) {
    x + y;
}
```

```
$ rustc functions.rs
functions.rs:1:14: 5:15 error: expected one of `:` or `@`, found `,`
functions.rs:1 fn add(x, y) {
                         ^
```

Functions

```rust
fn add(x, y) {
    x + y;
}
```

```
$ rustc functions.rs
functions.rs:1:14: 5:15 error:
functions.rs:1 fn add(x, y) {
                        ^
```

expected one of `:` or `@`, found `,`

Functions

```
fn add(x: i32, y: i32) {
    x + y;
}
```

Functions

```
fn add(x: i32, y: i32) {
    x + y;
}
```

Functions

```
fn add(x: i32, y: i32) {
    x + y
}
```

Functions

```rust
fn add(x: i32, y: i32) {
    x + y
}
```

```
functions.rs:2:5: 2:10 error: mismatched types:
 expected `()`,
    found `i32`
(expected (),
    found i32) [E0308]
functions.rs:2     x + y
                   ^~~~~
functions.rs:2:5: 2:10 help: run `rustc --explain E0308` to see a detailed explanation
error: aborting due to previous error
```

Functions

```rust
fn add(x: i32, y: i32) {
    x + y
}
```

```
functions.rs:2:5: 2:10 error: mismatched types:
 expected `()`,
    found `i32`
(expected (),
    found i32) [E0308]
functions.rs:2     x + y
                   ^~~~~
functions.rs:2:5: 2:10 help: run `rustc --explain E0308` to see a detailed explanation
error: aborting due to previous error
```

Functions

```
fn add(x: i32, y: i32) -> i32 {
    x + y
}
```

Functions

```
fn add(x: i32, y: i32) -> i32 {
    x + y
}
```

Functions

```
fn add(x: i32, y: i32) -> i32 {
    x + y;
}
```

Functions

```
fn add(x: i32, y: i32) -> i32 {
    x + y;
}
```

Expressions
Vs.
Statements

Variable Binding in Ruby

```
a = b = 'c'
```

Variable Binding in Rust

```
let x = (let y = '😱');
```

Variable Binding in Rust

```
let x = (y = '😱');
```

Methods

```rust
struct Circle {
    x: f64,
    y: f64,
    radius: f64,
}

impl Circle {
    fn area(&self) -> f64 {
            std::f64::consts::PI * (self.radius * self.radius)
        }
}

fn main() {
    let c = Circle { x: 0.0, y: 0.0, radius: 2.0 };
    println!("{}", c.area());
}
```

"Strings"

&str Vs. String

&str Vs. String

(Symbol Vs. String)

medium.com/@mfpiccolo/a-rubyist-rusting-ii-f72dd8b0ed97

Collections

Ruby

Arrays and Hashes

Rust!

Vec

VecMap

LinkedList

BinaryHeap

BitSet

HashSet

BitVec

BTreeSet

BTreeMap

HashMap

VecDeque

Vec

VecMap

LinkedList

BitSet

BinaryHeap

HashSet

BitVec

BTreeSet

BTreeMap

HashMap

VecDeque

Array Vs. Vector

Iterating Vectors

```
let vec = vec![1, 2, 3, 4];
for x in vec.iter() {
    println!("vec contained {}", x);
}
```

Iterating Vectors

```
let mut vec = vec![1, 2, 3, 4];
for x in vec.iter_mut() {
    *x += 1;
}
```

Pattern Matching

Pattern Matching

```
struct Point {
    x: i32,
    y: i32,
}

fn main() {
    let point = Point { x: 1, y: 1 };

    match point {
        Point { x, y } if x == y => println!("X and Y are equal!"),
        _                        => println!("Sorry, not equal"),
    }
}
```

Why should I care?

Ruby S*CK in Concurrency

Compile Error!

```
use std::thread;

fn main() {
    let mut data = vec![1, 2, 3];

    for i in 0..3 {
            thread::spawn(move || {
                        data[i] += 1;
                    });
        }

    thread::sleep_ms(50);
}
```

Compile Error!

```
intro.rs:8:25: 8:29 error: capture of moved value: `data` [E0382]
intro.rs:8                                         data[i] += 1;
                                                   ^~~~
```

Compile Error!

```
intro.rs:8:25: 8:29 error: capture of moved value: `data` [E0382]
intro.rs:8                 data[i] += 1;
                           ^~~~
```

Ruby is Slow

Microservices

Obrigado